(초안)

한글의 자모 조합에 기반한 생성형 텍스트의 워터마킹 설계

Robust Korean Text Watermarking based on Jamo-Combination and Generation-Time Biasing

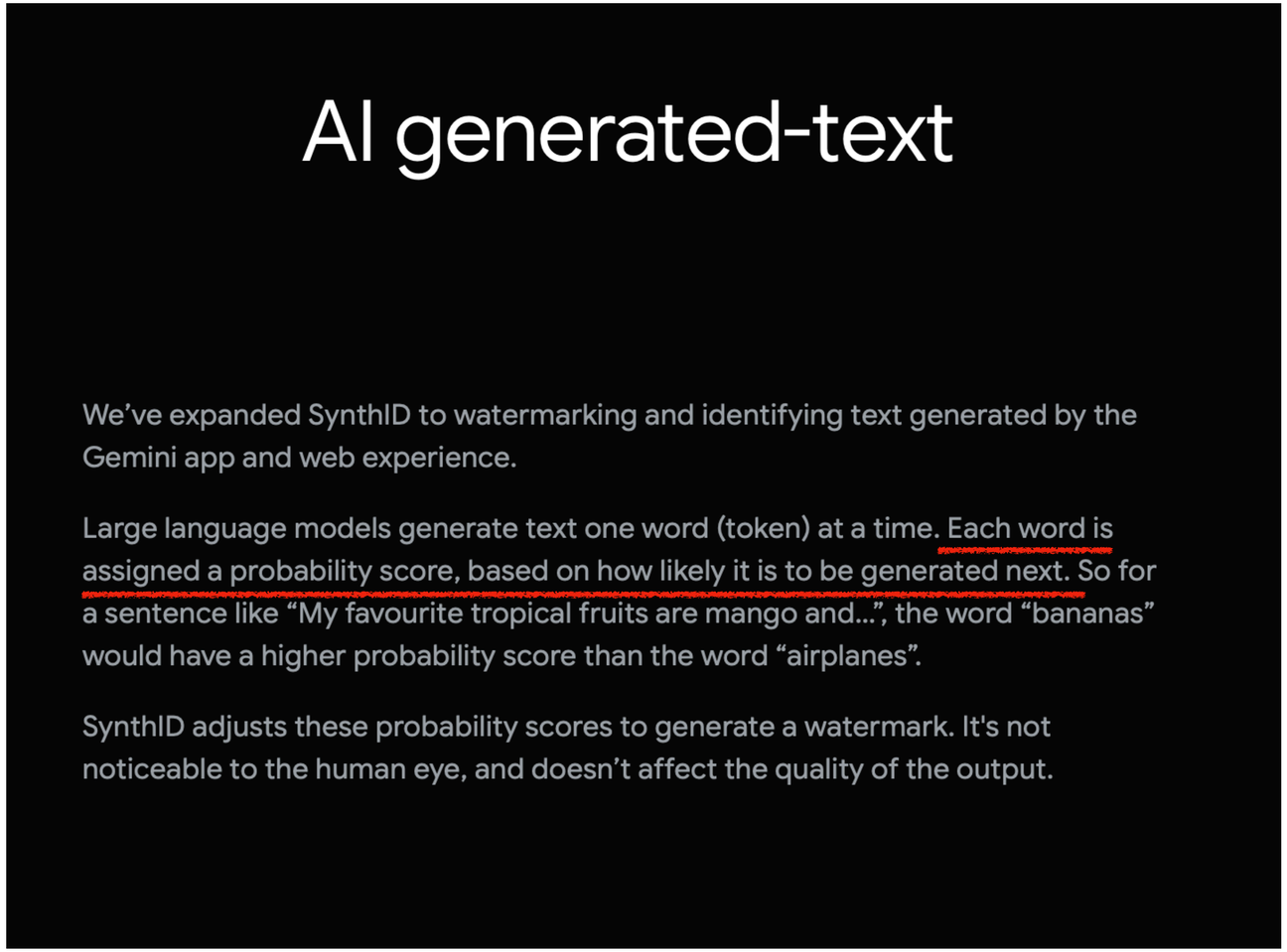
강희지 (컴퓨터과학과)

목차

1. 연구 제안

1-1. 배경

- 기존 워터마킹 방식



이미지출처: SynthID (Google DeepMind)

1. 연구 제안

1-2. 목적

- 한글의 언어적 특성을 이용한 워터마크 삽입 및 탐지 방법 설계
- 기존 영어 기반 워터마킹 방식과 비교하여 분석

- 초성 19자
  예) ㄱ, ㄴ, ㄷ, ㄹ, ㄲ, ㄸ, ㅃ, …
- 중성 21자
  예) ㅏ, ㅖ, ㅣ, ㅗ, ㅜ, …
- 종성 27자
  예) ㄱ, ㄴ, ㄹㄱ, ㄴㅈ, …

  → 한글 자모 단위의 **높은 조합성**을 이용하여
  자연스러우면서도 **많은 정보를 삽입**할 수 있는 **한국어 특화 워터마크**를 설계

1. 연구 제안

1-3. 구상

- 텍스트 생성(generation-time) 단계에서, 토큰 분해
- 자모 단위로 정보값을 매핑

- 토큰을 자모 단위로 표현한다는 가정 하에,
  - 한글의 언어적 특성: 초성/중성/종성 세 종류의 토큰
  - 조합을 통해 다양한 음절 생성 → 상대적으로 다양한 토큰의 고른 분포
  - 기대 결과
    - 높은 엔트로피 값
    - 다음 토큰 예측 불확실성

2. 사전 학습

2.1 기존 LLM에서 한글의 토큰화 방식

- Gemini

  토큰 정보

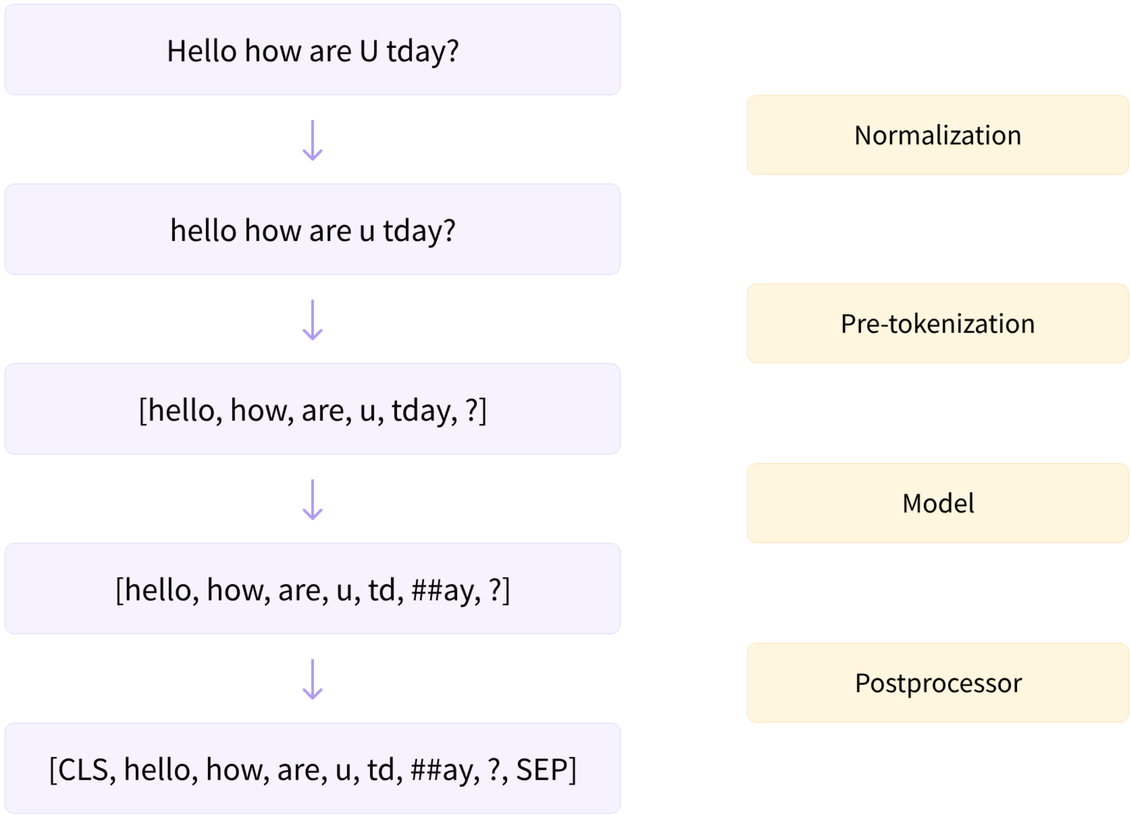
  토큰은 단일 문자(예: `z`) 또는 전체 단어(예: `cat`)일 수 있습니다. 긴 단어는 여러 토큰으로 나뉩니다. 모델에서 사용하는 모든 토큰 집합을 어휘라고 하며, 텍스트를 토큰으로 분할하는 프로세스를 *토큰화*라고 합니다.

  Gemini 모델의 경우 토큰은 약 4자와 같습니다. 토큰 100개는 영어 단어 약 60~80개에 해당합니다.

- LLaMA

  - The LLaMA tokenizer is a BPE model based on sentencepiece. One quirk of sentencepiece is that when decoding a sequence, if the first token is the start of the word (e.g. “Banana”), the tokenizer does not prepend the prefix space to the string.

- BPE (Byte Pair Encoding)
- WordPiece
- Unigram
- Etc.

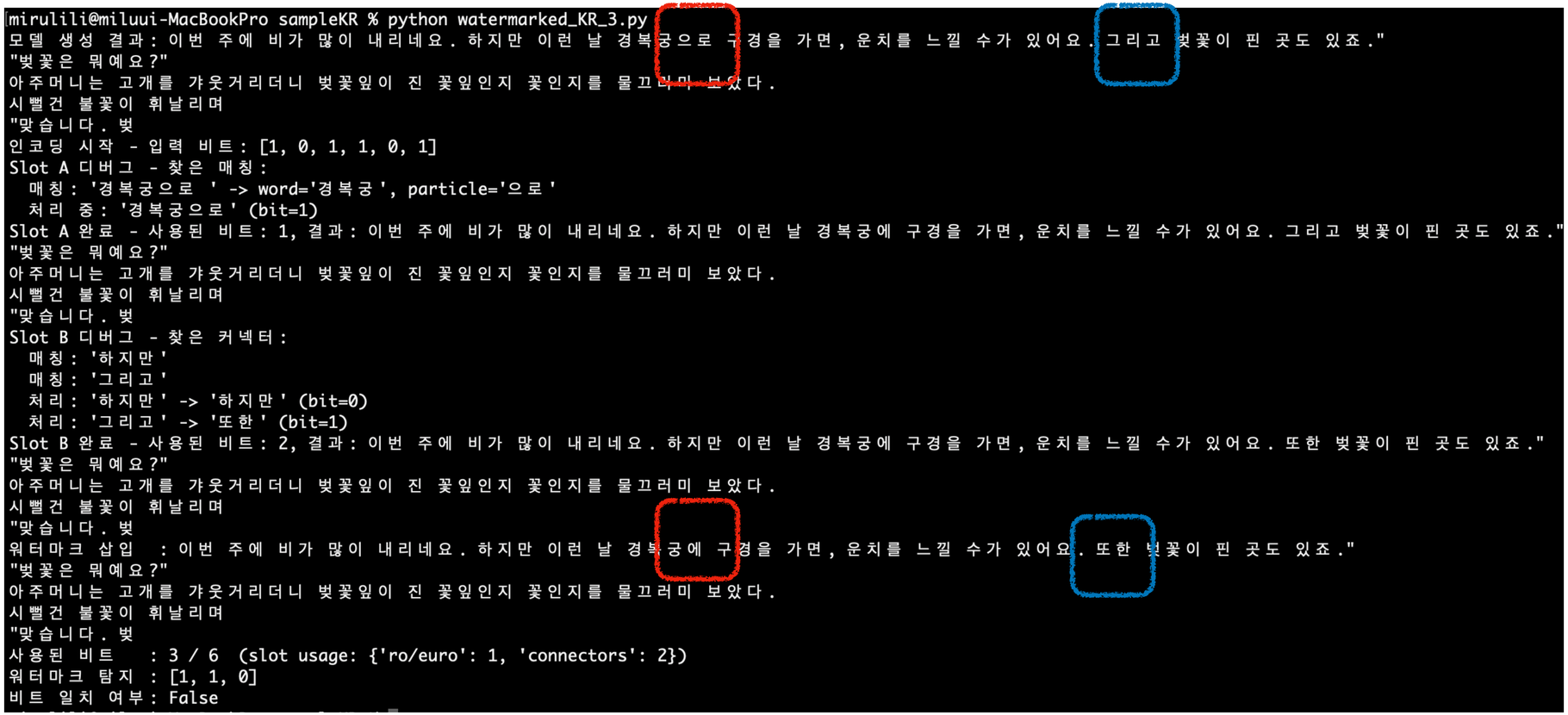


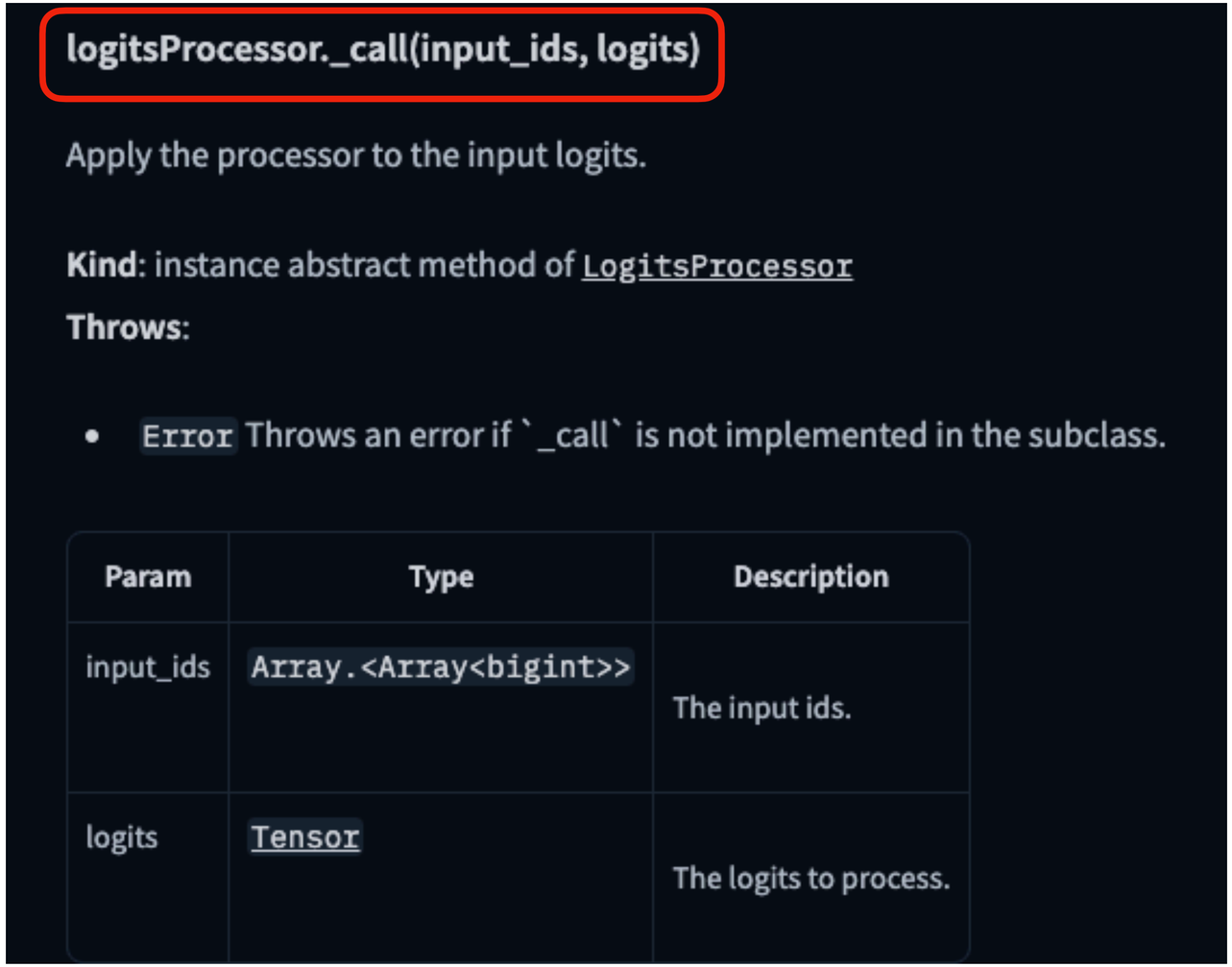
이미지출처: HuggingFace

2. 사전 학습

2-2. Gerneration-time 워터마킹

- 간단한 실험 진행





logitsProcessor._call(input_ids, logits)

Apply the processor to the input logits.

Kind: instance abstract method of `LogitsProcessor`

Throws:

- `Error` Throws an error if `_call` is not implemented in the subclass.

Param	Type	Description
input_ids	`Array.<Array<bigint>>`	The input ids.
logits	`Tensor`	The logits to process.

이미지출처: HuggingFace

convert_ids_to_tokens

<source>

```
( ids: typing.Union[int, list[int]], skip_special_tokens: bool = False ) → str or list[str]
```

Parameters

- **ids** (`int` or `list[int]`) — The token id (or token ids) to convert to tokens.
- **skip_special_tokens** (`bool`, *optional*, defaults to `False`) — Whether or not to remove special tokens in the decoding.

Returns `str` or `list[str]`

The decoded token(s).

Converts a single index or a sequence of indices in a token or a sequence of tokens, using the vocabulary and added tokens.

decode

<source>

```
( token_ids: typing.Union[int, list[int], ForwardRef('np.ndarray'), ForwardRef('torch.Tensor'), ForwardRef('tf.Tensor')], skip_special_tokens: bool = False, clean_up_tokenization_spaces: typing.Optional[bool] = None, **kwargs ) → str
```

Parameters

- **token_ids** (`Union[int, list[int], np.ndarray, torch.Tensor, tf.Tensor]`) — List of tokenized input ids. Can be obtained using the `__call__` method.
- **skip_special_tokens** (`bool`, *optional*, defaults to `False`) — Whether or not to remove special tokens in the decoding.
- **clean_up_tokenization_spaces** (`bool`, *optional*) — Whether or not to clean up the tokenization spaces. If `None`, will default to `self.clean_up_tokenization_spaces`.
- **kwargs** (additional keyword arguments, *optional*) — Will be passed to the underlying model specific decode method.

Returns `str`

The decoded sentence.

Converts a sequence of ids in a string, using the tokenizer and vocabulary with options to remove special tokens and clean up tokenization spaces.

Similar to doing `self.convert_tokens_to_string(self.convert_ids_to_tokens(token_ids))`.

이미지출처: HuggingFace

3. 실험 설계

3.1 접근 방법

- 실험 방향
  - Generation-time 자모 워터마크 필터
  - 토크나이저 변경 불필요

- LogitsProcessor 단계에서 후보 토큰들의 자모 분해 (유니코드 산술분해*)
- 분해 후 계산한 비트 값에 따라 조건에 맞는 토큰 선택
- 탐지:
  - 생성된 텍스트 자모 분해
  - 비트 패턴 통계로 검출

3. 실험 설계

3.2 사용 모델 및 데이터

- 모델: GPT2-small, KoGPT2-small 등 경량 모델
- 데이터: 온라인 상 한국어 코퍼스

3. 실험 설계

3.3 구현 흐름

- 후보 토큰 t의 음절 s에 대해, 초성, 중성, 종성(x, y, z)으로 분해
- 예시) 해시 함수 hash(s) = (x + y + z) mod n
- 주어진 비트 값 b
- 후보 토큰 t의 음절 집합 U(s) 중에서, hash(s) = b를 만족시키면, 이 토큰을 선택

4. Trade-off

- Semantic 변형 공격

5. 기대 결과 및 의의

- 한글 구조적 기반의 generation-time 워터마킹 시도
- 영어식 방식과 유사한 품질 유지하며 정보 capacity 및 탐지 성능에 차별성

6. 참고 문헌

- Dathathri et al. (2024). *Scalable watermarking for identifying large language model outputs.*
- Gibney, E. (2024). *Google unveils invisible ‘watermark’ for AI-generated text*. Nature. https://www.nature.com/articles/d41586-024-03462-7
- Google DeepMind. synthid-text. GitHub. https://github.com/google-deepmind/synthid-text
- Google DeepMind. SynthID website. https://deepmind.google/science/synthid/
- Google. Gemini API. https://ai.google.dev/gemini-api/docs/tokens
- Google, SentencePiece. GitHub. https://github.com/google/sentencepiece
- Hugging Face. LLM Course: Chapter 6. https://huggingface.co/learn/llm-course/chapter6/
- Hugging Face. Introducing SynthID Text. Hugging Face Blog. https://huggingface.co/blog/synthid-text
- Kirchenbauer et al. (2023). *A Watermark for Large Language Models.*
- Park et al. (2020). *An Empirical Study of Tokenization Strategies for Various Korean NLP Tasks.*
- Hugging Face. LLaMA. Transformers documentation. https://huggingface.co/docs/transformers/v4.42.4/model_doc/llama
- OpenAI. tiktoken. GitHub. https://github.com/openai/tiktoken
- Bryce et al. “딥 러닝을 이용한 자연어 처리 입문”. https://wikidocs.net/book/2155